

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

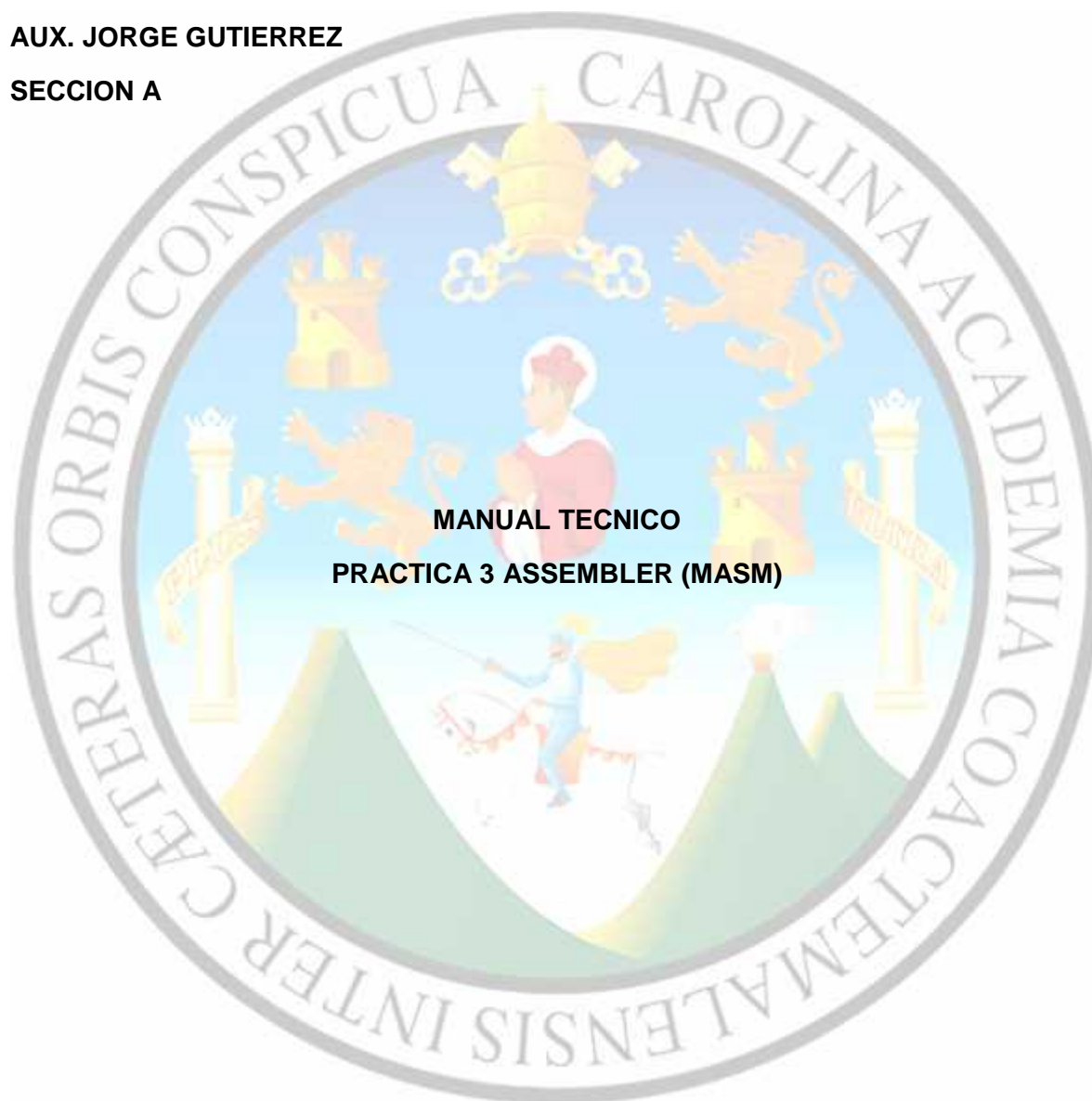
CIENCIAS Y SISTEMAS

ARQUITECTURA DE COMPUTADORES 1

ING. OTTO RENE LEIVA

AUX. JORGE GUTIERREZ

SECCION A



MANUAL TECNICO
PRACTICA 3 ASSEMBLER (MASM)

NOMBRE: OSCAR RENE CUELLAR MANCILLA

CARNET: 201503712

FECHA: 3 DE OCTUBRE DE 2017

ENSAMBLADOR USADO EN LA PRACTICA: (MASM)

El Microsoft Macro Assembler (MASM) es un ensamblador para la familia x86 de microprocesadores. Fue producido originalmente por Microsoft para el trabajo de desarrollo en su sistema operativo MS-DOS, y fue durante cierto tiempo el ensamblador más popular disponible para ese sistema operativo. El MASM soportó una amplia variedad de facilidades para macros y programación estructurada, incluyendo construcciones de alto nivel para bucles, llamadas a procedimientos y alternación (por lo tanto, MASM es un ejemplo de un ensamblador de alto nivel). Versiones posteriores agregaron la capacidad de producir programas para los sistemas operativos Windows. MASM es una de las pocas herramientas de desarrollo de Microsoft para las cuales no había versiones separadas de 16 bits y 32 bits.

CÓDIGO RELEVANTE DE LA PRACTICA:

Durante la realización de la práctica se utilizaron varios macros, el más utilizado sería el macro llamado "print" con el cual mando como parámetro una cadena a imprimir, se manda al registro AX el @data que representa que se va a escribir una cadena, con el mov ah,09 le indico a mi interrupción 21H que iniciare una impresión de cadena en pantalla y al registro DX le mando la dirección donde se almacena mi cadena a imprimir

```
print macro cadena
    mov ax,@data
    mov ds,ax
    mov ah,09
    mov dx,offset cadena
    int 21h
endm
```

Se declararon todos los mensajes que podían ser mostrados en consola con variables que los representaban de tipo byte:

Cada uno de los cuales es llamado en diferente ocasión y con diferente propósito.

[illegible]

OBTENER UN NUMERO LEIDO DESDE CONSOLA:

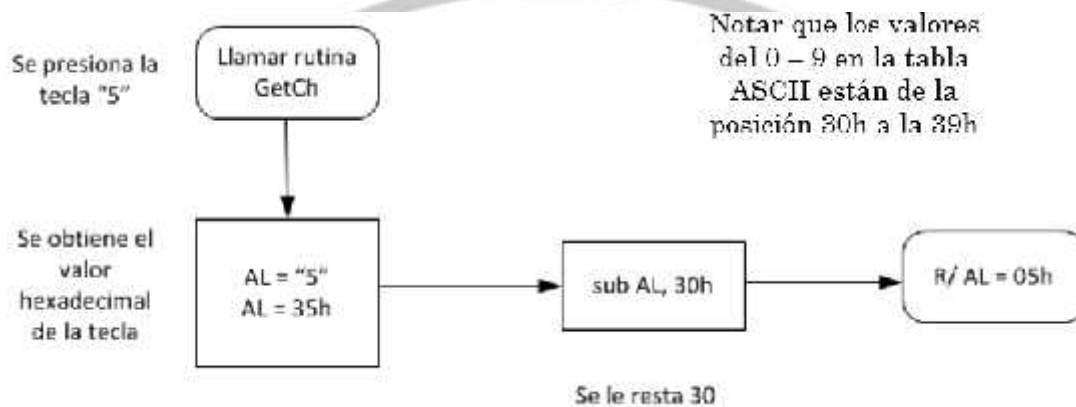
El siguiente algoritmo pide que se ingrese un numero el cual es guardado en un arreglo de tipo byte, el cual posteriormente será mandado al algoritmo de conversión a decimal. Este algoritmo permite el ingreso de signo solo al principio, se podrán ingresar números con el signo + o - solo al comienzo de lo contrario mostrara una advertencia y volverá a solicitar el número, si no se ingresa nada y se ingresa un salto de línea únicamente, este guardara un 0 por default, si se ingresan caracteres incorrectos también se mostrará una advertencia.

```
-----GUARDAR NUMERO-----  
;---GUARDA UN NUMERO CON SIGNO, PERMITE EL SIGNO + O - SOLO AL PRINCIPIO  
ObtenerNumero macro buffer,lim  
LOCAL ON2,Signo2,valido2,FinON2,NoCero  
xor cx,cx  
xor si,si  
  
ON2:  
cmp cx,lim  
je Error2 ;error cuando ingresa mas de los caracteres soportados  
mov ah,01h  
int 21h  
cmp al,0dh ;ascii del \n  
je FinON2  
cmp al,30h ;ascii del 0  
jb Signo2  
cmp al,39h ;ascii del 9  
ja Error1  
jmp valido2  
  
Signo2:  
cmp cx,0  
jne Error1 ;error cuando vuelve a ingresar un + o - luego del inicio  
cmp al,2bh ;ascii del +  
je valido2  
cmp al,2dh ;ascii del -  
je valido2  
jmp Error1  
  
valido2:  
mov buffer[si],al  
inc si  
inc cx  
jmp ON2  
  
FinON2:  
cmp cx,0 ;si no ingreso nada, solo el enter, asigno un 0 default  
jne NoCero  
mov buffer[si],30h  
NoCero:  
xor si,si  
endm
```

CONVERTIR UN NUMERO A DECIMAL:

Al obtener un numero desde consola se pide sea ingresado carácter por carácter, en el siguiente algoritmo se le resta un 30h que es la diferencia entre los números y sus códigos ascii para poder tenerlo en decimal, este número se guarda en una variable de tipo Word. A continuación, se muestra el diagrama de flujo de la conversión y el algoritmo ya implementado en el código:

DIAGRAMA DE FLUJO DE LA CONVERSION:



ALGORITMO IMPLEMENTADO EN EL CODIGO:

```
ConvertToInt macro num,buffer
    int al,finNumero,saveNum,omitirMas,complementoA2
    mov si,si
    mov cx,cx
    mov ax,ax
    mov dx,dx
    mov bx,bx
    mov dl,10

    mov buffer[si],0ah ;ascii del +
    je OmitirMas
    cmp buffer[si],2dh ;ascii del +
    jnc saveNum

    mov cx,1 ;bandera para saber si tengo que sacar complemento o no
OmitirMas:
    inc si ;paso al primer numero

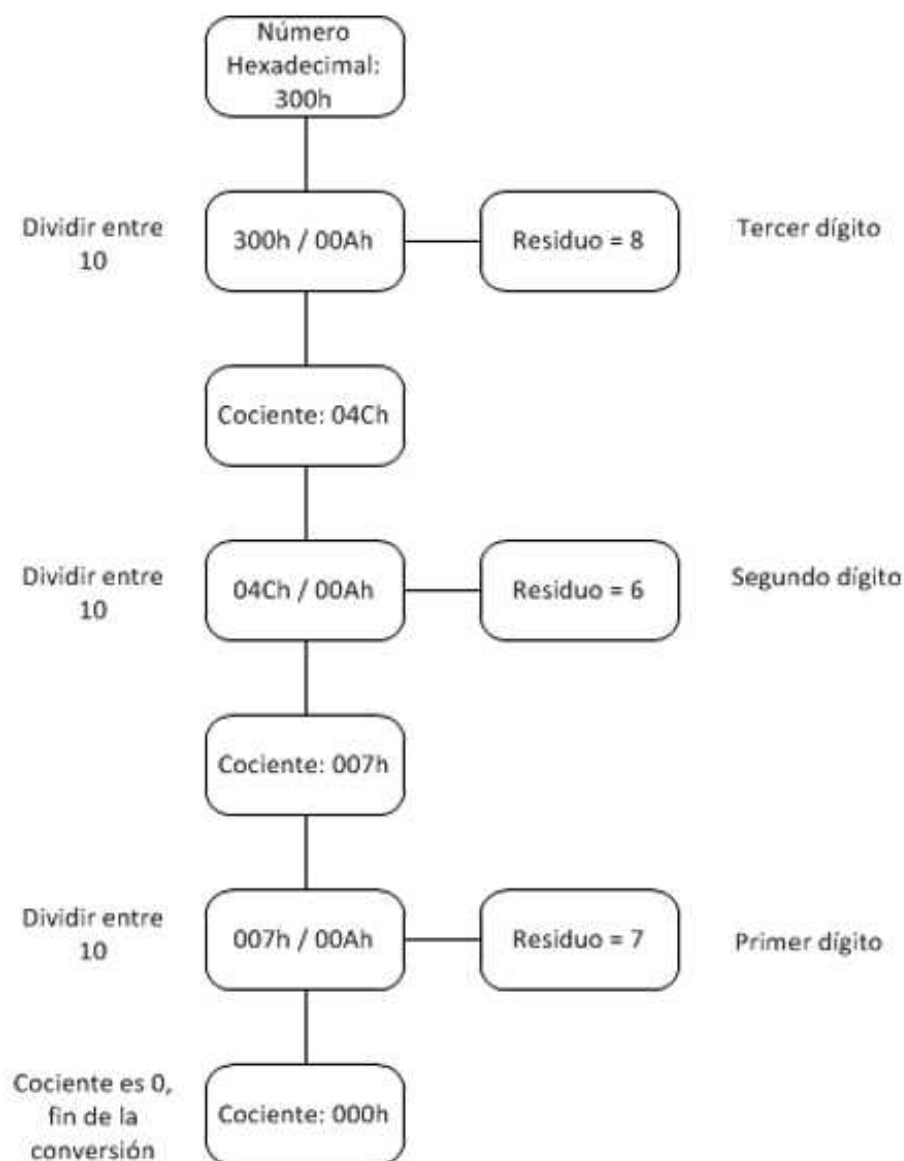
saveNum:
    ;print errr2
    mov bl,buffer[si] ;muevo el ascii del primer numero al registro bl
    sub bl,30h ;le resto 30h que es la diferencia que hay entre los numeros y sus respectivas ascii
    inc si ;paso al siguiente digito
    add al,bl ;sumo lo que llevo en AL contra lo que acabo de encontrar
    cmp buffer[si],20h ;ascii del espacio en blanco
    je finNumero
    cmp buffer[si],24h ;ascii del $
    je finNumero
    mul dl ;multiplico el valor que llevo en AL contra 10
    jmp saveNum

finNumero:
    cmp cx,1
    jnc NoComplementoA2
    neg ax
NoComplementoA2:
    mov num,ax
    ;print num
endm
```

CONVERTIR UN DECIMAL A SUS CODIGOS ASCIIS:

Por el lado contrario, cuando realizamos operaciones con números decimales y necesitamos mostrarlos en pantalla hay que hacer un procedimiento inverso para pasar cada dígito del número a su respectivo código ascii y así poder mostrarlo en pantalla. Para poder mostrar el resultado en pantalla se utiliza el algoritmo de convertir un número entero a sus respectivos ascis, les sumo 30h que es la diferencia que hay entre los números y sus códigos ascis.

DIAGRAMA DE FLUJO DE LA CONVERSION:



ALGORITMO IMPLEMENTADO:

```

;-----CONVERTIR RESULTADO-----
ConvertirResultado macro res,buffer
xor si,si
xor cx,cx
xor ax,ax
xor dx,dx
mov ax,res
mov dl,0ah
jmp segDiv

segDiv1:
xor ah,ah
segDiv:
div dl
;print msg10
inc cx
push ax
cmp al,00h ;si ya dio 0 en el cociente dejar de dividir
je FinCR
jmp segDiv1

FinCR:
pop ax
add ah,30h
mov buffer[si],ah
inc si
loop FinCR

mov ah,24h ;ascii del $
mov buffer[si],ah
inc si
endm

```

CREACION DEL REPORTE:

Para la creación del reporte se utilizaron los macros para crear archivos y modificar su contenido, para ello se utilizaron los siguientes macros:

```

crear macro ruta,handle
lea dx,ruta
mov ah,3ch
mov cx,00h
int 21h
mov handle,ax
jc Error7
endm

```

```

escribir macro numbytes,databuffer,handle
mov ah,40h
mov bx,handle
mov cx,numbytes
lea dx,databuffer
int 21h
jc Error9
endm

```

```

cerrar macro handle
mov ah,3eh
mov bx,handle
int 21h
jc Error11
endm

```

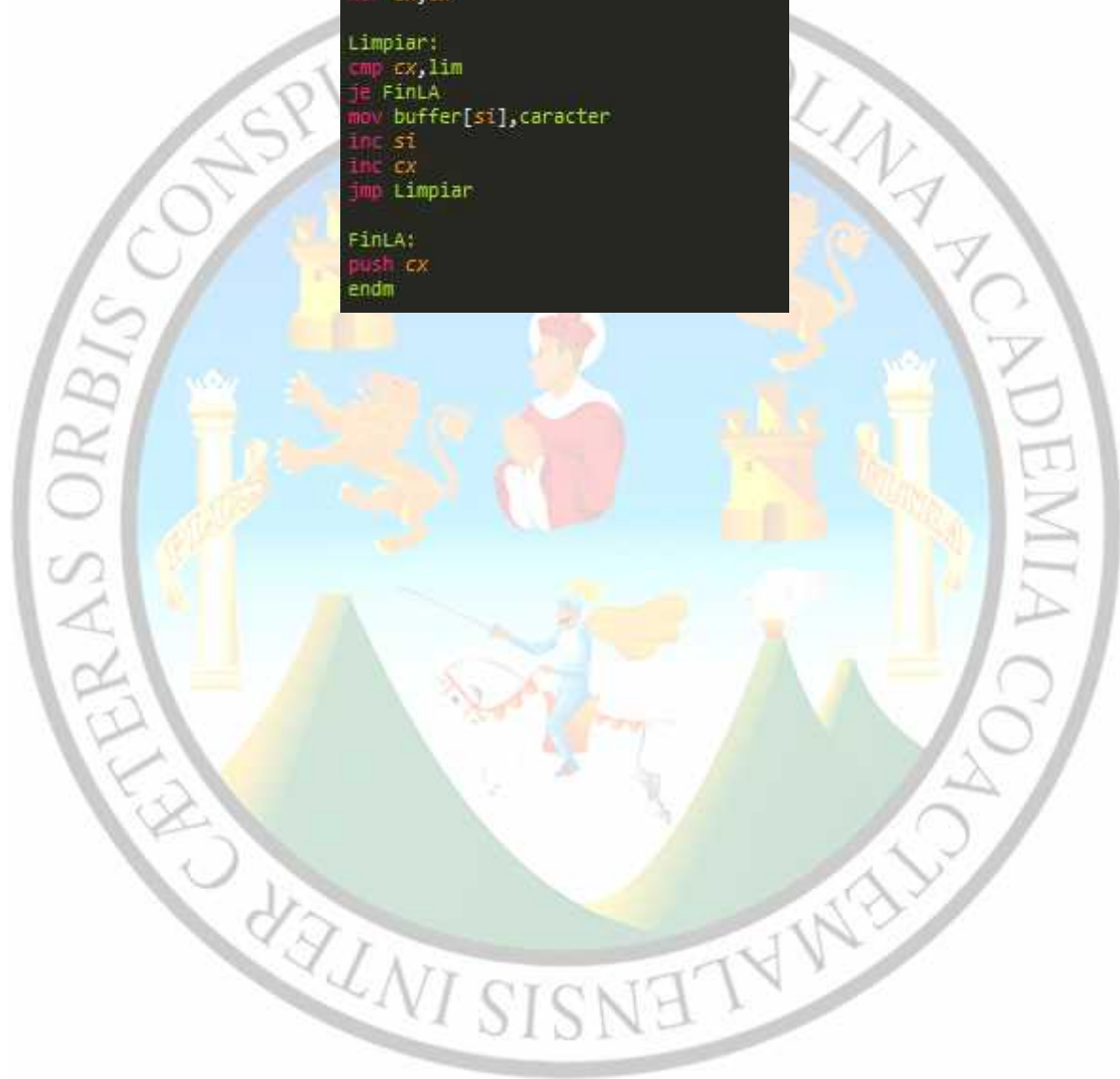
LIMPIAR ARREGLO:

Para que no se encuentren errores a la hora de reutilizar un arreglo con la basura digital, se utiliza el siguiente macro que limpia un arreglo de tipo byte con un carácter específico y la cantidad de espacios específicas.

```
LimpiarArr macro buffer,caracter,lim
LOCAL Limpiar, FinLA
pop cx
xor si,si
xor cx,cx

Limpiar:
cmp cx,lim
je FinLA
mov buffer[si],caracter
inc si
inc cx
jmp Limpiar

FinLA:
push cx
endm
```



REQUERIMIENTOS MINIMOS:

MACROS PARA GRAFICAR:

En los siguientes macros se utilizó la interrupción 10h, con su número de función 13h, el cual nos da el modo de video necesario para dibujar los respectivos pixeles.

```
;-----REGRESAR AL MODO DE TEXTO-----
ModoTexto macro
;Regresar a modo texto
mov ax,0003h
int 10h
endm

;-----MODO GRAFICO-----
ModoGrafico macro
;Iniciacion de modo video
mov ax, 0013h
int 10h
endm

;-----PINTAR PIXEL GRAFICO
pixel macro x0, y0, color
push cx
mov ah, 0ch
mov al, color
mov bh, 0h
mov dx, y0
mov cx, x0
int 10h
pop cx
endm

;-----Pintar eje X
PintarX macro
LOCAL eje_x
;Dibujar eje de las abscisas
mov cx,13eh
eje_x:
pixel cx,63h,4fh
loop eje_x
endm

;-----Pintar eje Y
PintarY macro cent
LOCAL eje_y
;Dibujar eje de las ordenadas
mov cx,0c6h
eje_y:
pixel cent,cx,4fh
loop eje_y
endm
```

El macro PintarY recibe como parámetro un centro, que es un número que está basado en los límites ingresados por el usuario, este sirve para colocar el eje y de tal manera que si se ingresa [-5,20] este centro estaría colocado más al lado izquierdo ya que no hay mucho que graficar desde la parte de -5.

GRAFICAR FUNCION NORMAL:

```
;-----GRAFIAR ORIGINAL-----
;GRAFIAR ORIGINAL
GraficarOriginal:
    ;LimpiarConsole
    PedirRango Lim0,Lim1,bufTemp,centro

    ;ModoGrafico
    PintarX
    PintarY centro
    EvaluarFuncion x0,x1,x2,x3,x4,cero,Lim0,Lim1,relacion,centro,y,x,resMayor,EsIntegral

    getch
    ModoTexto
    ;LimpiarConsole
    jmp Graficar
```

Para realizar todas las gráficas se utiliza el macro “EvaluarFuncion” el cual recibe como parámetro 5 constantes, los límites de la gráfica, la relación entre los límites y los 200 pixeles de ancho y las variables x, y que irán cambiando.

Este macro valúa las constantes elevadas a su potencia respectiva y las va sumando para así poder obtener una coordenada x,y que será la que se mande a graficar en el macro DibujarPixel.

```
;-----obtengo f(cx)-----
mov cx,5
Valuar c5,tamano,cx
mov cx,4
Valuar c4,tamano,cx
mov cx,3
Valuar c3,tamano,cx
mov cx,2
Valuar c2,tamano,cx
mov cx,1
Valuar c1,tamano,cx
mov cx,0
Valuar c0,tamano,cx ;resultado final de F(cx) queda en AX
```

GRAFICAR DERIVADA:

Se manda como parámetro las mismas variables que en la parte anterior a excepción de las constantes ya que están serán diferentes ya que la derivada elimina la constante y baja cada potencia de las demás variables x.

```
;-----GRAFIAR DERIVADA-----
;GRAFIAR DERIVADA
GraficarDerivada:
    ;LimpiarConsole
    PedirRango Lim0,Lim1,bufTemp,centro

    ;ModoGrafico
    PintarX
    PintarY centro
    EvaluarFuncion 0x0,0x1,0x2,0x3,cero,cero,Lim0,Lim1,relacion,centro,y,x,resMayor,EsIntegral

    getch
    ModoTexto
    ;LimpiarConsole
    jmp Graficar
```

En este apartado lo que cambia con respecto a las dos anteriores es que se solicita el ingreso de un numero al usuario el cual representara a la constante de integración. Este número se pide antes de realizar la gráfica y es enviado como x a la potencia 0 para que no sea multiplicado por ningún valor si no solamente sumado.

[illegible]

Interrupción al sistema 21H:

- Introducimos en (AH) el número de función a la que deseamos acceder.
- En caso de que deseamos acceder a una sub-función dentro de una función, debemos indicarlo introduciendo en (AL) el número de esa sub-función.
- Llamar a la INT 21H.

El modo gráfico 13h nos permite manejar la pantalla como una matriz de 320 píxeles de ancho por 200 píxeles de alto. Cada píxel puede tomar uno de 256 colores, estos colores están definidos en una paleta de colores la cual podemos configurar.

Usamos la interrupción 10h, servicio 0Ch para modificar los píxeles; el parámetro AL más que especificar directamente el color, indica la entrada de la paleta de colores que se debe usar para el píxel ubicado en la fila DX y la columna CX.